Particle Metropolis adjusted Langevin algorithms for state-space models

Christopher Nemeth and Paul Fearnhead

Lancaster University

February 5, 2014

Abstract

Particle MCMC is a class of algorithms that can be used to analyse state-space models. They use MCMC moves to update the parameters of the models, and particle filters to propose values for the path of the state-space model. Currently the default is to use random walk Metropolis to update the parameter values. We show that it is possible to use information from the output of the particle filter to obtain better proposal distributions for the parameters. In particular it is possible to obtain estimates of the gradient of the log posterior from each run of the particle filter, and use these estimates within a Langevin-type proposal. We propose using the recent computationally efficient approach of Nemeth et al. (2013) for obtaining such estimates. We show empirically that for a variety of state-space models this proposal is more efficient than the standard random walk Metropolis proposal in terms of: reducing autocorrelation of the posterior samples, reducing the burn-in time of the MCMC sampler and increasing the squared jump distance between posterior samples.

1 Introduction

Markov chain Monte Carlo (MCMC) algorithms are a popular and well studied methodology that can be used to draw samples from posterior distributions. MCMC has allowed Bayesian statistics to evolve beyond simple tractable models to more complex and realistic models where the posterior may only be known up to a constant of proportionality. Over the past few years MCMC methodology has been extended further to tackle problems where the model likelihood is intractable. For such models it is often possible to replace the intractable likelihood with an estimate (Beaumont, 2003), which can be obtained from Monte Carlo simulations. Andrieu and Roberts (2009) showed that within the MCMC sampler, if the likelihood is replaced with an unbiased estimate, then the sampler still targets the correct stationary distribution. Andrieu et al. (2010) extended this work further to create a class of MCMC algorithms for state-space models based on sequential Monte Carlo methods (also known as particle filters). This class of algorithms is referred to as particle MCMC. In this paper we shall focus on one particular algorithm, the particle marginal Metropolis Hastings

algorithm which replaces the likelihood term in the Metropolis Hastings (MH) sampler with an unbiased particle filter estimator.

In the standard Metropolis Hastings algorithm a popular proposal is the random walk Metropolis (RWM). This proposal selects new parameter values by perturbing the previous values with random Gaussian noise. The efficiency of the MH algorithm is dependent on the magnitude of the noise added. Theoretical results have established that tuning this proposal such that approximately 23.4% of the proposed samples are accepted is optimal as the number of parameters tend to infinity (Roberts et al., 1997). An extension to the RWM proposal is the Metropolis adjusted Langevin algorithm (MALA) which incorporates an estimate of the gradient of the posterior within the proposal distribution. This proposal has the advantage of steering the proposed parameters towards the mode of the posterior. Thus proposed values are more likely to be accepted, and it has an optimal acceptance rate of 57.4% (Roberts and Rosenthal, 1998).

As with the standard Metropolis Hastings algorithm the efficiency of the particle marginal Metropolis Hastings algorithm is also affected by the choice of proposal distribution. For state-space models, it is generally not possible to use the MALA proposal because, as with the likelihood, the gradient of the log posterior is intractable. In this paper we present an algorithm for creating approximations of the gradient of the log posterior using output from the particle filter, based on the algorithm given by Nemeth et al. (2013). The particle approximation of the gradient is then used within the MALA framework to create a new proposal which we refer to as particle MALA (pMALA).

In order for the pMALA algorithm to be practicable it is important that the extra computational cost of estimating the gradient of the log posterior is small, while the estimate itself is accurate. As such, the use of the algorithm from Nemeth et al. (2013) is central to our algorithm. This algorithm has a cost that is linear in the number of particles, and requires only a small overhead on top of running a standard particle filter. It has been shown to have a much smaller Monte Carlo error for estimating the gradient than other algorithms whose computational cost in linear in the number of particles. A similar pMALA algorithm has been independently proposed by Dahlin et al. (2013a), but their algorithm for estimating the gradient has a computational cost that is quadratic in the number of particles, and hence leads to a much slower pMALA algorithm.

The outline of the paper is as follows. We first give an introduction to state-space models, and to MCMC and sequential Monte Carlo (particle filter) algorithms for analysing these models. In Section 3 we introduce particle MCMC, and show that information from running the particle filter can be used to guide the choice of proposal distribution for the parameters. We then introduce our pMALA algorithm. Section 4 presents empirical results comparing pMALA and standard particle MCMC algorithms across a range of examples. The paper ends with a discussion.

2 Inference for state space models

2.1 State space models

Consider the general state space model where there is a latent Markov process $\{X_t; 1 \leq t \leq T\}$ that takes values on some measurable space $\mathcal{X} \subseteq \mathbb{R}^{n_x}$. The pro-

cess is fully characterised by its initial density $p(x_1|\theta) = \mu_\theta(x_1)$ and transition probability density

$$p(x_t|x_{1:t-1}, \theta) = p(x_t|x_{t-1}, \theta) = f_\theta(x_t|x_{t-1}),$$

where $\theta \in \Theta$ represents a vector of model parameters. For an arbitrary sequence $\{z_i\}$ the notation $z_{i:j}$ corresponds to $(z_i, z_{i+1}, \ldots, z_j)$ for $i \leq j$.

We assume that the process $\{X_t\}$ is not directly observable, but partial observations are received via a second process $\{Y_t; 1 \leq t \leq T\} \subseteq \mathcal{Y}^{n_y}$. The observations $\{Y_t\}$ are conditionally independent given $\{X_t\}$ and are defined by the probability density

$$p(y_t|y_{1:t-1}, x_{1:t}, \theta) = p(y_t|x_t, \theta) = g_\theta(y_t|x_t).$$

The marginal likelihood of observations for a given θ can be decomposed as

$$p(y_{1:T}|\theta) = p(y_1|\theta) \prod_{t=2}^{T} p(y_t|y_{1:t-1}, \theta), \tag{1}$$

where,

$$p(y_t|y_{1:t-1}, \theta) = \int g_\theta(y_t|x_t) \int f_\theta(x_t|x_{t-1}) p(x_{t-1}|y_{1:t-1}, \theta) dx_{t-1} dx_t$$

is the predictive likelihood.

Aside from a few special cases, it is generally not possible to evaluate the likelihood analytically, but it is often possible to approximate the likelihood using importance sampling (Pitt, 2002). Model parameters θ can be estimated by maximising the likelihood using expectation maximisation (EM) or gradient based maximum likelihood methods (Nemeth et al., 2013; Poyiadjis et al., 2011; Dempster et al., 1977). Alternatively, within the Bayesian framework, MCMC techniques (Andrieu et al., 2010; Fearnhead, 2011) can be applied to estimate the posterior density $p(\theta|y_{1:T})$ of the parameters conditional on the observed data. Within this paper we shall consider only the latter case of applying MCMC to state-space models.

2.2 MCMC for state-space models

We start by considering the generic MCMC algorithm used to perform Bayesian inference on the parameters θ. Firstly, we introduce a prior distribution for the parameters, $p(\theta)$. Our goal is then to estimate the posterior density $p(\theta|y_{1:T}) \propto p(y_{1:T}|\theta)p(\theta)$, which is known only up to a constant of proportionality. In this setting we are considering the ideal case, where we assume that the likelihood (1) is known and tractable.

Samples from the posterior $(\theta_1, \theta_2, \ldots, \theta_j, \ldots, \theta_J)$ are generated using the Metropolis Hastings algorithm where proposed values θ' are sampled from a proposal distribution $q(\cdot|\theta_{j-1})$ and accepted (i.e. $\theta_j = \theta'$) with probability

$$\alpha(\theta'|\theta_{j-1}) = \min\left\{1, \frac{p(y_{1:T}|\theta')p(\theta')q(\theta_{j-1}|\theta')}{p(y_{1:T}|\theta_{j-1})p(\theta_{j-1})q(\theta'|\theta_{j-1})}\right\}. \tag{2}$$

The samples $\{\theta_j\}_{j=1}^J$ generated by the MH algorithm form a Markov chain of correlated samples. The choice of proposal distribution $q(\theta'|\theta)$ is important as it

affects the autocorrelation of the samples from the algorithm. A standard choice of proposal is the Gaussian random walk proposal. This proposal generates new parameter values by perturbing the current parameters with noise sampled from a Gaussian distribution with zero mean and covariance matrix Σ. The covariance matrix can be chosen to account for correlations in the parameter vector θ. Or in the simplest case, each element of θ is perturbed independently by replacing the covariance matrix with $\sigma_\epsilon^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and σ_ϵ^2 is a user chosen step size parameter. For this simple case, new parameters θ' are sampled from

$$\theta' = \theta_{j-1} + \sigma_\epsilon z \quad \text{where} \quad z \sim \mathcal{N}(0, \mathbf{I}).$$

The efficiency of the Gaussian random walk proposal is determined by the scaling of the step size parameter. Theoretical results show that as the number of parameters $d \to \infty$ the optimal acceptance rate for the MH ratio (2) is 0.234 (Roberts et al., 1997). A great deal of research has been dedicated to the optimal scaling of the Gaussian random walk proposal and the interested reader is referred to Roberts and Rosenthal (2001) for a review.

Alternatively, efficient proposal distributions can be designed using the geometry of the posterior density (Roberts and Tweedie, 1996; Girolami and Calderhead, 2011) to improve the mixing of the MCMC sampler. One such approach is the Metropolis adjusted Langevin algorithm (Roberts and Rosenthal, 1998) which uses the gradient of the log posterior $\nabla \log p(\theta|y_{1:T})$ within the proposal

$$\theta' = \theta_{j-1} + \sigma_\epsilon z + \frac{\sigma_\epsilon^2}{2}\nabla \log p(\theta_{j-1}|y_{1:T})$$

where $z \sim \mathcal{N}(0, \mathbf{I})$ and σ_ϵ is the step size. The gradient of the log posterior can be given in terms of the score vector (gradient of the loglikelihood) $\nabla \log p(y_{1:T}|\theta)$ and the gradient of the log prior density $\nabla \log p(\theta|y_{1:T}) = \nabla \log p(y_{1:T}|\theta) + \nabla \log p(\theta)$.

Samples proposed using MALA tend to be less correlated compared to samples generated from the Gaussian random walk proposal. Intuitively, this is because using the gradient of the log posterior steers the proposed samples towards the mode of the posterior allowing for more ambitious jumps in the parameter space which are likely to be accepted. Roberts and Rosenthal (1998) showed that applying MALA within the MCMC sampler gives an optimal acceptance rate of 0.574, much higher than the standard Gaussian random walk proposal. Also they show that the mixing of MALA scales much better with dimension than random walk Metropolis.

The outline of MCMC given above is appropriate for the idealised scenario where the likelihood $p(y_{1:T}|\theta)$ is tractable. However, for most state-space models this is not the case. Andrieu and Roberts (2009) showed that by replacing the likelihood with a Monte Carlo estimate $\hat{p}(y_{1:T}|\theta)$, which is non-negative and unbiased, the MCMC sampler will still target the correct posterior distribution. One way of obtaining unbiased estimates of the likelihood for state-space models is to use a particle filter.

2.3 Sequential Monte Carlo

Sequential Monte Carlo algorithms represent a class of simulation methods for the sequential approximation of posterior probability distributions. In the con-

text of state-space modelling, we are interested in approximating the posterior $p(x_t|y_{1:t}, \theta)$ of the filtered latent state x_t, given a sequence of observations $y_{1:t}$. In this section we shall assume that the model parameters θ are fixed. Approximations of $p(x_t|y_{1:t}, \theta)$ can be calculated recursively by first approximating $p(x_1|y_1, \theta)$, then $p(x_2|y_{1:2}, \theta)$ and so forth for $t = 1, \ldots, T$. At time t the posterior of the filtered state is

$$p(x_t|y_{1:t}, \theta) \propto g_\theta(y_t|x_t) \int f_\theta(x_t|x_{t-1}) p(x_{t-1}|y_{1:t-1}, \theta) dx_{t-1} \tag{3}$$

where $p(x_{t-1}|y_{1:t-1}, \theta)$ is the posterior at time $t-1$.

The posterior at time t can be approximated if we assume that at time $t-1$ we have a set of particles $\{x_{t-1}^{(i)}\}_{i=1}^N$ and corresponding weights $\{w_{t-1}^{(i)}\}_{i=1}^N$ which produce a discrete approximation of $p(x_{t-1}|y_{1:t-1}, \theta)$. The Monte Carlo approximation for (3) at time t is then

$$\hat{p}(x_t|y_{1:t}, \theta) \approx c g_\theta(y_t|x_t) \sum_{i=1}^{N} w_{t-1}^{(i)} f_\theta(x_t|x_{t-1}^{(i)}), \tag{4}$$

where c is a normalising constant. The particle approximation $\hat{p}(x_t|y_{1:t}, \theta)$ tends to the true density $p(x_t|y_{1:t}, \theta)$ as the number of particles $N \to \infty$ (Crisan and Doucet, 2002). The filtered density, as given above, can be updated recursively by propagating and updating the particle set using importance sampling techniques. The resulting algorithms are called particle filters, see Doucet et al. (2000) and Cappé et al. (2007) for a review.

In this paper the particle approximations of the latent process are created with the auxiliary particle filter of Pitt and Shephard (1999). This filter can be viewed as a general filter from which simpler filters are given as special cases. We shall consider the version of this filter as presented in Fearnhead et al. (2010). The aim is to view the target (4) as defining a joint distribution on the particle at time $t-1$ and the value of a new particle at time t. The probability of sampling particle $x_{t-1}^{(i)}$ and using a conditional density for then sampling x_t is

$$c w_{t-1}^{(i)} g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}^{(i)}).$$

We approximate this with $\xi_t^{(i)} q(x_t|x_{t-1}^{(i)}, y_t, \theta)$, where $q(x_t|x_{t-1}^{(i)}, y_t, \theta)$ is a density function that can be sampled from and $\{\xi_t^{(i)}\}_{i=1}^N$ are a set of probabilities. This defines a proposal which we can simulate from by first choosing particle $x_{t-1}^{(i)}$ with probability $\xi_t^{(i)}$, and then, conditional on this, a new particle value, x_t, is sampled from $q(x_t|x_{t-1}^{(i)}, y_t, \theta)$. The weight assigned to our new particle is then

$$\tilde{w}_t = \frac{w_{t-1}^{(i)} g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}^{(i)})}{\xi_t^{(i)} q(x_t|x_{t-1}^{(i)}, y_t, \theta)}.$$

Details are summarised in Algorithm 1.

Simpler filters, such as the bootstrap filter (Gordon et al., 1993), can be derived from the general filter by setting $q(x_t|x_{t-1}^{(i)}, y_t, \theta) = f_\theta(x_t|x_{t-1}^{(i)})$ and $\xi_t^{(i)} = w_{t-1}^{(i)}$. The bootstrap filter is a popular choice due to its simplicity, however, this filter can be inefficient as it does not take account of the newest

observations in the proposal, and therefore can lead to the propagation of particles that are likely to be given small weights.

The optimal proposal density, in terms of minimising the variance of the weights (Doucet et al., 2000), is available when $q(x_t|x_{t-1}^{(i)}, y_t, \theta) = p(x_t|x_{t-1}^{(i)}, y_t, \theta)$ and $\xi_t^{(i)} \propto w_{t-1}^{(i)} p(y_t|x_{t-1}^{(i)})$. This filter is said to be *fully adapted* as all the weights $w_t^{(i)}$ will equal $1/N$. Generally it is not possible to sample from the optimal proposal, but alternative proposals can be used which approximate the fully adapted filter.

Algorithm 1 Auxiliary Particle Filter

Step 1: Iteration $t = 1$,

(a) For $i = 1, \ldots, N$, sample particles $\{x_1^{(i)}\}$ from the prior $p(x_1|\theta)$ and set $\tilde{w}_1^{(i)} = p(y_1|x_1^{(i)})$.

(b) Calculate $C_1 = \sum_{i=1}^N \tilde{w}_1^{(i)}$; set $\hat{p}(y_1) = C_1/N$; and calculate normalised weights $w_1^{(i)} = \tilde{w}_1^{(i)}/C_1$ for $i = 1, \ldots, N$.

Step 2: Iteration $t = 2, \ldots, T$. Assume a user-defined set of proposal weights $\{\xi_t^{(i)}\}_{i=1}^N$ and family of proposal distributions $q(x_t|x_{t-1}^{(i)}, y_t, \theta)$.

(a) Sample indices $\{k_1, k_2, \ldots, k_N\}$ from $\{1, \ldots, N\}$ with probabilities $\xi_t^{(i)}$.

(b) Propagate particles $x_t^{(i)} \sim q(\cdot|x_{t-1}^{(k_i)}, y_t, \theta)$.

(c) Weight particles $\tilde{w}_t^{(i)} = \frac{w_{t-1}^{(k_i)} g_\theta(y_t|x_t^{(i)}) f_\theta(x_t^{(i)}|x_{t-1}^{(k_i)})}{\xi_t^{(k_i)} q(x_t^{(i)}|x_{t-1}^{(k_i)}, y_t, \theta)}$ and calculate $C_t = \sum_{i=1}^N \tilde{w}_t^{(i)}$.

(d) Obtain an estimate of the predictive likelihood, $\hat{p}(y_t|y_{1:t-1}, \theta) = C_t/N$, and calculate normalised weights $w_t^{(i)} = \tilde{w}_t^{(i)}/C_t$ for $i = 1, \ldots, N$.

One of the benefits of using the particle filter is that an estimate for the likelihood $p(y_{1:T}|\theta)$ is given for free from the particle filter output. We can estimate $p(y_t|y_{1:t-1}, \theta)$ by

$$\hat{p}(y_t|y_{1:t-1}, \theta) = \left[\sum_{i=1}^N \frac{\tilde{w}_t^{(i)}}{N}\right], \tag{5}$$

where, $\tilde{w}_t^{(i)}$ are unnormalised weights. An unbiased estimate of the likelihood is then

$$\hat{p}(y_{1:T}|\theta) = \hat{p}(y_1|\theta) \prod_{t=2}^T \hat{p}(y_t|y_{1:t-1}, \theta).$$

See Algorithm 1, and Pitt et al. (2012) and Del Moral (2004) for further details.

3 Particle MCMC

3.1 Particle marginal Metropolis Hastings

The auxiliary particle filter given in Algorithm 1 provides a positive, unbiased estimate of the likelihood based on the importance weights (5). Andrieu and Roberts (2009) and Andrieu et al. (2010) have shown how we can use such estimates in place of the likelihood function within MCMC. The idea is to run

Algorithm 1 at each iteration of an MCMC algorithm to get an estimate of the likelihood for the current parameter value. We then use this estimate instead of the true likelihood value within the accept-reject probability. If interest lies just in the posterior for the parameter, this results in the particle marginal Metropolis Hastings (PMMH) algorithm (see Algorithm 2). We will focus on this algorithm in the following (see Andrieu et al., 2010, for alternative particle MCMC algorithms).

Algorithm 2 Particle Marginal Metropolis Hastings (PMMH) Algorithm

Step 1: iteration $j = 1$,
(a) Set θ_1 arbitrarily.
(b) Run Algorithm 1 and compute the marginal likelihood $\hat{p}(y_{1:T}|\theta_1)$ from the importance weights (5).
Step 2: iteration $j = 2, \ldots, M$.
(a) Sample $\theta' \sim q(\cdot|\theta_{j-1})$
(b) Run Algorithm 1 and compute the marginal likelihood $\hat{p}(y_{1:T}|\theta')$ from the importance weights (5).
(c) Set $\theta_j = \theta'$ and $\hat{p}(y_{1:T}|\theta_j) = \hat{p}(y_{1:T}|\theta')$
with probability $1 \wedge \frac{\hat{p}(y_{1:T}|\theta')p(\theta')q(\theta_{j-1}|\theta')}{\hat{p}(y_{1:T}|\theta_{j-1})p(\theta_{j-1})q(\theta'|\theta_{j-1})}$
else set $\theta_j = \theta_{j-1}$ and $\hat{p}(y_{1:T}|\theta_j) = \hat{p}(y_{1:T}|\theta_{j-1})$.

A key result is that PMMH has $p(\theta|y_{1:T})$ as its stationary distribution (Andrieu and Roberts, 2009; Andrieu et al., 2010). Let $\mathcal{U}$ denote the random variables used in the particle filter to generate the estimate of the likelihood, and $p(\mathcal{U}|\theta)$ their conditional density given θ. We can define a target distribution on $(\theta,\mathcal{U})$ which is

$$\hat{p}(\theta,\mathcal{U}|y_{1:T}) \propto \hat{p}(y_{1:T}|\theta,\mathcal{U})p(\mathcal{U}|\theta)p(\theta). \tag{6}$$

It is straightforward to show that PMMH is a standard MCMC algorithm with this target distribution, and with a proposal distribution $q(\theta'|\theta)p(\mathcal{U}|\theta')$. Furthermore, the marginal target distribution for θ is just

$$\begin{aligned}\int \hat{p}(\theta,\mathcal{U}|y_{1:T})d\mathcal{U} &\propto \int \hat{p}(y_{1:T}|\theta,\mathcal{U})p(\mathcal{U}|\theta)p(\theta)d\mathcal{U} \\ &= p(y_{1:T}|\theta)p(\theta),\end{aligned}$$

the posterior, $p(\theta|y_{1:T})$. The last equality follows from the fact that $\hat{p}(y_{1:T}|\theta,\mathcal{U})$ is unbiased estimator of the likelihood. Note that implementation of PMMH does not require storing all details of the particle filter, $\mathcal{U}$, just the resulting estimate of the likelihood $\hat{p}(y_{1:T}|\theta,\mathcal{U})$.

Whilst PMMH admits $p(\theta|y_{1:T})$ as the invariant density regardless of the variance of the likelihood estimator $\hat{p}(y_{1:T}|\theta,\mathcal{U})$, the variance does affect the mixing properties of the algorithm; see Pitt et al. (2012) and Sherlock et al. (2013) for details. The choice of proposal distribution for the parameter, $q(\cdot|\theta)$, will also have an important impact on the mixing properties of the algorithm. We now show that information from the particle filter can be used to guide this choice of proposal.

3.2 Efficient use of the particle filter output

Consider using some information from the particle filter, which we will denote $\mathcal{I}(\mathcal{U})$, within the proposal distribution. So if the current state of the Markov chain is $(\theta,\mathcal{U})$, our proposal will be $\hat{q}(\theta'|\theta,\mathcal{I}(\mathcal{U}))$. The acceptance probability of a new state $(\theta',\mathcal{U}')$ will then be

$$\alpha(\theta',\mathcal{U}'|\theta,\mathcal{U}) = \min\left\{1, \frac{\hat{p}(y_{1:T}|\theta',\mathcal{U}')p(\theta')\hat{q}(\theta|\theta',\mathcal{I}(\mathcal{U}'))}{\hat{p}(y_{1:T}|\theta,\mathcal{U})p(\theta)\hat{q}(\theta'|\theta,\mathcal{I}(\mathcal{U}))}\right\}. \tag{7}$$

It is straightforward to show that such an algorithm admits $p(\theta|y_{1:T})$ as the invariant density :

Proposition 3.1 *Implementing PMMH with proposal distribution $\hat{q}(\theta'|\theta,\mathcal{I}(\mathcal{U}))$, and acceptance probability given by (7), gives an MCMC algorithm which admits $p(\theta|y_{1:T})$ as the invariant density.*

Proof As before the PMMH is a standard MCMC algorithm with $\hat{p}(\theta,\mathcal{U}|y_{1:T})$ as its invariant distribution, but now the proposal distribution is $\hat{q}(\theta'|\theta,\mathcal{I}(\mathcal{U}))p(\mathcal{U}'|\theta')$. The acceptance probability for such an MCMC algorithm is

$$\alpha(\theta',\mathcal{U}'|\theta,\mathcal{U}) = \min\left\{1, \frac{\hat{p}(y_{1:T}|\theta',\mathcal{U}')p(\mathcal{U}'|\theta')p(\theta')\hat{q}(\theta|\theta',\mathcal{I}(\mathcal{U}'))p(\mathcal{U}|\theta)}{\hat{p}(y_{1:T}|\theta,\mathcal{U})p(\mathcal{U}|\theta)p(\theta)\hat{q}(\theta'|\theta,\mathcal{I}(\mathcal{U}))p(\mathcal{U}'|\theta')}\right\}$$

which simplifies to (7) as required. □

Again, when implementing this version of PMMH we do not need to store all details of the particle filter. All we need is to store our estimate of the likelihood $\hat{p}(y_{1:T}|\theta,\mathcal{U})$ and the information $\mathcal{I}(\mathcal{U})$.

Our choice of information will be an estimate of the score, $\mathcal{I}(\mathcal{U}) = \nabla \log \hat{p}(y_{1:T}|\theta)$, where we give details of how to obtain such an estimate in the next section. We then use this estimate in place of the true score within a MALA proposal:

$$\hat{q}(\theta'|\theta,\mathcal{I}(\mathcal{U})) = \mathcal{N}\left(\theta_{j-1} + \frac{\sigma_\epsilon^2}{2}\left[\nabla \log \hat{p}(\theta|y_{1:T})\right], \sigma_\epsilon^2\right), \tag{8}$$

where $\nabla \log \hat{p}(\theta|y_{1:T}) = \nabla \log \hat{p}(y_{1:T}|\theta) + \nabla \log p(\theta)$, and σ_ϵ is the step-size parameter. The new proposal (8) can be used in place of $q(\theta'|\theta)$ in Algorithm 2 to give the particle MALA algorithm.

3.3 Particle approximations of the score vector

We can create a particle approximation of the score vector based on Fisher's identity (Cappé et al., 2005)

$$\begin{aligned}\nabla \log p(y_{1:T}|\theta) &= \int \nabla \log p(x_{1:T}, y_{1:T}|\theta) p(x_{1:T}|y_{1:T},\theta) dx_{1:T} \\ &= \mathbb{E}[\nabla \log p(x_{1:T}, y_{1:T}|\theta)|y_{1:T},\theta]\end{aligned}$$

which is the expectation of

$$\nabla \log p(x_{1:T}, y_{1:T}|\theta) = \nabla \log p(x_{1:T-1}, y_{1:T-1}|\theta) + \nabla \log g_\theta(y_T|x_T) + \nabla \log f_\theta(x_T|x_{T-1})$$

over the path $x_{1:T}$.

The particle approximation to the score vector is obtained by replacing $p(x_{1:T}|y_{1:T},\theta)$ with a particle approximation $\hat{p}(x_{1:T}|y_{1:T},\theta)$. Here we outline this idea, but see Poyiadjis et al. (2011) for more details.

For each particle at a time $t-1$, there is an associated path, defined by tracing the ancestry of each particle back in time. With slight abuse of notation denote this path by $x^{(i)}_{1:t-1}$. We can thus associate with particle i at time $t-1$ a value $\alpha^{(i)}_{t-1} = \nabla \log p(x^{(i)}_{1:t-1}, y_{1:t-1}|\theta)$. These values can be updated recursively. Remember that in step 2(b) of Algorithm 1 we sample k_i, which is the index of the particle at time $t-1$ that is propagated to produce the ith particle at time t. Thus we have

$$\alpha^{(i)}_t = \alpha^{(k_i)}_{t-1} + \nabla \log g_\theta(y_t|x^{(i)}_t) + \nabla \log f_\theta(x_t|x^{(k_i)}_{t-1}). \tag{9}$$

The problem with this approach is that the variance of the score estimate $\nabla \log p(y_{1:t}|\theta)$ increases quadratically with t (Poyiadjis et al., 2011) due to degeneracy in the approximation of α_t. Poyiadjis et al. (2011) suggest an alternative particle filter algorithm, which avoids a qudratically increasing variance but at the expense of a computational cost that is quadratic in the number of particles. Instead we will use the algorithm of Nemeth et al. (2013), which uses kernel density estimation and Rao-Blackwellisation to substantially reduce the Monte Carlo variance, but still maintains an algorithm whose computational cost is linear in the number of particles.

An outline of their approach is as follows. We first use kernel density estimation to replace each discrete $\alpha^{(i)}_{t-1}$ value by a Gaussian distribution:

$$\alpha^{(i)}_{t-1} \sim \mathcal{N}(m^{(i)}_{t-1}, V_{t-1}). \tag{10}$$

The mean of this distribution is obtained by shrinking $\alpha^{(i)}_{t-1}$ towards the mean of α_{t-1},

$$m^{(i)}_{t-1} = \lambda \alpha^{(i)}_{t-1} + (1-\lambda) \sum_{i=1}^{N} w^{(i)}_{t-1} \alpha^{(i)}_{t-1}.$$

Here $0 < \lambda < 1$ is a user-defined shrinkage parameter. The idea of this shrinkage is that it corrects for the increase in variability introduced through the kernel density estimation of West (1993). For a definition of V_{t-1} see Nemeth et al. (2013), however its actual value does not affect the following details.

The resulting model for the α_ts, including their updates (9), is linear Gaussian. Hence we can use Rao-Blackwellisation to avoid sampling the $\alpha^{(i)}_t$s, and instead calculate the parameters of the kernel (10) directly. This gives the following recursion for the means,

$$\begin{aligned} m^{(i)}_t &= \lambda m^{(k_i)}_{t-1} + (1-\lambda) \sum_{i=1}^{N} w^{(i)}_{t-1} m^{(i)}_{t-1} \\ &\quad + \nabla \log g_\theta(y_t|x^{(i)}_t) + \nabla \log f_\theta(x^{(i)}_t|x^{(k_i)}_{t-1}). \end{aligned} \tag{11}$$

The final score estimate depends only on these means, and is

$$\nabla \log \hat{p}(y_{1:t}|\theta) = \sum_{i=1}^{N} w^{(i)}_t m^{(i)}_t.$$

Algorithm 3 Rao-Blackwellised Kernel Density Estimate of the Score Vector

Add the following steps to Algorithm 1.

Step 1:

(c) Set $\nabla \log \hat{p}(y_1|\theta) = \nabla \log g_\theta(y_1|x_1^{(i)}) + \nabla \log \mu_\theta(x_1^{(i)})$.

Step 2:

(e) For $i = 1, \ldots, N$, calculate

$$\begin{aligned} m_t^{(i)} &= \lambda m_{t-1}^{(k_i)} + (1-\lambda) \sum_{i=1}^{N} w_{t-1}^{(i)} m_{t-1}^{(i)} \\ &\quad + \nabla \log g_\theta(y_t|x_t^{(i)}) + \nabla \log f_\theta(x_t^{(i)}|x_{t-1}^{(k_i)}). \end{aligned}$$

(f) Update and store the score vector

$$\nabla \log \hat{p}(y_{1:t}|\theta) = \sum_{i=1}^{N} w_t^{(i)} m_t^{(i)}.$$

See Algorithm 3 for a summary.

When $\lambda = 1$ the recursion simplifies to the method given by Poyiadjis et al. (2011), where the variance of the score estimate will increase quadratically with t. The use of a shrinkage parameter $\lambda < 1$ alleviates the degeneracy problems that affect the estimation of the score and significantly reduces the estimate's variance. As a rule of thumb, setting $\lambda = 0.95$ produces reliable estimates where the variance of the score estimate increases only linearly with t (see Nemeth et al. (2013) for further details). We shall use this tuning for all examples given in the Section 4.

4 Simulation Studies

In this section we compare the particle marginal Metropolis Hastings algorithm, using the random walk Metropolis proposal (10), which we shall refer to as PMMH, against the particle MALA proposal (8). The two proposals shall be compared in terms of the their inefficiency, which is measured by the integrated autocorrelation time of the Markov chain, Ineff $= 1 + 2\sum_{m=1}^{\infty} \rho_l$, where ρ_l is the autocorrelation of the Markov chain at lag l. The infinite sum in the integrated autocorrelation time is truncated to L^*, which is the lag after which the autocorrelations are approximately zero. As a rule of thumb the maximum number of lags $L^* = \min\{1000, L\}$, where L is the lowest index for l such that $|\rho_l| < 2/\sqrt{M}$ and M is the sample size used to compute ρ_l. Lower values for the inefficiency indicate less correlation between samples, see Pitt et al. (2012) for further details of this metric.

The MCMC algorithms can also be compared using the squared jump distance

$$\text{SJD} = \frac{1}{M-1} \sum_{m=1}^{M} |\theta_{m+1} - \theta_m|^2.$$

This metric measures the average distance between successive posterior samples, where larger jumps correspond to better mixing of the MCMC sampler and improved exploration of the posterior.

All results are given as the average of 10 independent Monte Carlo simulations, where for each simulation the PMMH algorithm (Alg. 2) is run for $100,000$ iterations. Only the last $50,000$ iterations are taken as samples from the posterior with the first $50,000$ iterations treated as burn-in.

4.1 Linear Gaussian Model

We start by considering the linear Gaussian state-space model, where is it is possible to estimate the marginal likelihood $p(y_{1:T}|\theta)$ and score vector exactly with the Kalman filter (Durbin and Koopman, 2001). This model provides a benchmark for comparing the efficiency of PMMH and pMALA. We also implement the MH and MALA algorithms using the exact estimates of the likelihood and score vector given by the Kalman filter. Finally, a comparison is also given for both the $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ algorithms of Poyiadjis et al. (2011) to estimate the score vector. Proposals created using the $\mathcal{O}(N^2)$ algorithm have been implemented by Dahlin et al. (2013a).

Consider the following linear Gaussian model

$$\begin{aligned} y_t &= \alpha + \beta x_t + \tau\epsilon_t, \\ x_t &= \mu + \phi x_{t-1} + \sigma\eta_t, \\ x_0 &\sim \mathcal{N}(\mu/(1-\phi), \sigma^2/(1-\phi^2)), \end{aligned}$$

where ϵ_t and η_t are standard independent Gaussian random variables and $\theta = (\alpha, \beta, \tau, \mu, \phi, \sigma)$ are model parameters.

For this model it is possible to use the fully adapted particle filter using the optimal proposal for the latent states (see Appendix A for details). Compared to the simpler bootstrap filter this will reduce the variance of the weights, which will therefore reduce the variance of the likelihood estimate.

We use simulated data from the model where 500 observations are generated with model parameters $\alpha = 0.2$, $\beta = 1$, $\tau = 1$, $\mu = 0.1$, $\phi = 0.9$, $\sigma = 0.15$. At each iteration of the PMMH/pMALA algorithm an estimate of the likelihood and score vector was calculated from the particle filter (Alg. 1 and 3) using 500 and 2000 particles. For the Poyiadjis $\mathcal{O}(N^2)$ algorithm, the particle filter is run with $\sqrt{N}$ particles to match the computational cost of the PMMH and pMALA algorithms.

The MCMC sampler was run with the following prior distributions

$$\begin{pmatrix}\alpha \\ \beta\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}0.3 \\ 1.2\end{pmatrix}, \tau^2 \begin{pmatrix}0.25 & 0 \\ 0 & 0.5\end{pmatrix}\right), \tau^2 \sim \mathcal{IG}(1, 7/20),$$

$\mu \sim \mathcal{N}(0.15, 0.5)$, $(\phi + 1)/2 \sim Beta(20, 5)$ and $\sigma^2 \sim \mathcal{IG}(2, 1/40)$, where $\mathcal{IG}$ is an inverse gamma distribution.

The parameters (ϕ, σ, τ) are constrained such that $|\phi| < 1$, $\sigma > 0$ and $\tau > 0$. These parameters are transformed for the MCMC sampler as $\tanh(\phi)$, $\log(\sigma)$ and $\log(\tau)$, noting that this transformation now introduces a Jacobian term into the MH acceptance ratio (2).

As discussed in Section 2.2 the optimal acceptance rate for the standard random walk Metropolis algorithm is 0.234 as the number of parameters $d \to \infty$. Recent results given by Sherlock et al. (2013) show that for MH algorithms where the likelihood is replaced with an unbiased estimate, the optimal acceptance

rate is approximately 0.07. For the PMMH algorithm with RWM proposal we shall use the scaling suggested by the authors $\sigma_\epsilon^2 = (2.562)^2\Sigma/d$, where Σ is an estimate of the posterior covariance of the parameters θ given from a pilot run (we use 2.38 in place of 2.562 for the Kalman RWM). The particle MALA and Kalman MALA algorithms were scaled as $\sigma_\epsilon^2 = \Sigma/d^{1/3}$ to match the mixing rate of MALA algorithms (Roberts and Rosenthal, 1998).

Algorithm	Particles	Acc. rate	Inefficiency	
			Min	Max
Kal. RWM		0.13	52.47	78.31
Kal. MALA		0.25	27.33	51.55
PMMH	2000	0.14	59.02	107.87
	500	0.13	52.28	116.83
pMALA	2000	0.25	26.87	47.58
	500	0.24	29.65	58.91
Poy. $\mathcal{O}(N)$	2000	0.25	31.62	58.41
	500	0.12	30.26	61.71
Poy. $\mathcal{O}(N^2)$	$\sqrt{2000}$	0.16	50.05	111.67
	$\sqrt{500}$	0.12	128.20	170.97

Table 1: Linear Gaussian example. Comparison of the efficiency of PMMH, pMALA, Poyiadjis MALA and the exact estimates of the likelihood and score vector from the Kalman filter. Particle approximations are based on 500 and 2000 particles.

Table 1 summarises the results of the MCMC simulations where for ease of presentation we have presented the minimum and maximum inefficiencies for each algorithm over all parameters. The inefficiency of all particle filter based samplers is increased, and the acceptance rate decreased, when the number of particles is reduced. The sampler still targets the correct stationary distribution, but less efficiently as a smaller number of particles leads to an increase in the variance of the estimate of the likelihood. This increased inefficiency would be more noticeable for models where it is not possible to use the fully adapted importance proposal distribution. Increasing the number of the particles reduces the variance of the likelihood estimate and increases the acceptance rate of the MCMC sampler (Pitt et al., 2012).

The pMALA algorithm has an increased acceptance rate compared to PMMH and also displays reduced inefficiency. The estimate of the gradient of the log posterior which is used in the proposal allows for greater jumps in the posterior, which leads to reduced autocorrelation between posterior samples. The Poyiadjis $\mathcal{O}(N^2)$ implementation is less efficient than the pMALA algorithm when compared with equal computational effort. The increase in inefficiency of the $\mathcal{O}(N^2)$ algorithm is caused by an increase in the variance of both the likelihood and score estimate. This is caused by the reduced number of particles used to run the particle filter at equal computational cost to pMALA.

Table 2 gives the inefficiencies for pMALA and the Poyiadjis $\mathcal{O}(N)$ algorithm compared with equal computational cost. The pMALA algorithm is more efficient than the Poyiadjis $\mathcal{O}(N)$ implementation of MALA as the length of the data T increases. This is due to the increased variance in the score vector estimate given by the Poyiadjis $\mathcal{O}(N)$ algorithm, which has been proven to increase

Algorithm	Observations	Inefficiency	
		Min	Max
pMALA	5000	343.48	1036.37
	2000	76.98	360.07
	1000	44.02	58.93
Poy. $\mathcal{O}(N)$	5000	394.74	1208.75
	2000	95.16	403.37
	1000	45.65	59.86

Table 2: Linear Gaussian example. Comparison of the efficiency of pMALA and Poyiadjis $\mathcal{O}(N)$ MALA. Particle approximations are based on 500 particles over datasets of length $T = 1000, 2000, 5000$.

quadratically with T (see Poyiadjis et al. (2011) for details). The improved efficiency of pMALA is the result of the reduced Monte Carlo error in the score vector given by the Rao-Blackwellised score estimate (Algorithm 3). Proposition 3.1 establishes that arbitrary output from the particle filter can be used within the proposal, but as the pMALA proposal demonstrates, the efficiency of the sampler can be improved by choosing better proposals. For the example considered here, pMALA is up to 20% more efficient.

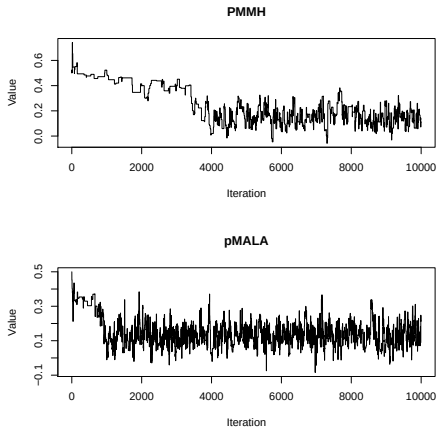

Figure 1: Linear Gaussian example. Trace plots of PMMH and pMALA for μ parameter.

The pMALA algorithm also has the advantage of reducing the burn-in time of the MCMC sampler. Consider the trace plot of the first $10,000$ samples of the parameter μ shown in Figure 1. The MCMC sampler using PMMH reaches stationarity after approximately $4,000$ iterations, where the chain is initially sticky with few new parameters accepted. Whereas pMALA reaches stationarity with less than $1,000$ iterations, given the same starting values for both samplers. Using information about the posterior (i.e. the gradient of the log posterior) pMALA can quickly reach stationarity and therefore significantly reduce the burn-in time of the MCMC sampler.

4.2 GARCH with noisy observations

This example considers the GARCH(1,1) model (Bollerslev et al., 1994) which has been extensively applied to financial returns data. We assume that the observations are observed with Gaussian noise

$$\begin{aligned} y_t &= x_t + \tau\epsilon_t, \quad x_t = \sigma_t^2\eta_t, \\ \sigma_t^2 &= \alpha + \beta x_{t-1}^2 + \gamma\sigma_{t-1}^2, \\ x_0 &\sim \mathcal{N}(0, \alpha/(1-\beta-\gamma)), \end{aligned}$$

where ϵ_t and η_t are standard independent Gaussian random variables and $\theta = (\alpha, \beta, \gamma, \tau)$ are the model parameters.

A dataset with 500 observations is sampled from the model using the parameters $\alpha = 0.1$, $\beta = 0.8$, $\gamma = 0.05$ and $\tau = 0.3$. The model parameters are estimated using the PMMH algorithm and compared against the pMALA implementation. Estimates of the likelihood and score vector, in the case of pMALA, are obtained from the particle filter using 1000 particles.

The parameters of this model must satisfy the following constraints: $\alpha > 0$, $\beta > 0$, $\gamma > 0$, $\tau > 0$ and $\beta + \gamma < 1$. These constraints can be satisfied by reparameterising the model so that $\phi = \alpha + \beta$, $\mu = \alpha/(1-\phi)$ and $\lambda = \beta/\phi$. The MCMC scheme is then completed by setting the prior distributions for the parameters: $(\phi+1)/2 \sim Beta(10, 3/2)$, $\mu \sim U(0, 2)$, $(\lambda+1)/2 \sim Beta(20, 3/2)$ and $\tau^2 \sim \mathcal{IG}(2, 1/2)$. Finally, the parameters are transformed to the unconstrained scale $\text{logit}(\phi)$, $\log(\mu)$, $\text{logit}(\lambda)$ and $\log(\tau)$ where the appropriate Jacobian is included in the MH ratio.

The proposal of the PMMH algorithm is scaled as $\sigma_\epsilon^2 = (2.562)^2(0.23, 1.43, 0.58, 0.011)/4$, where the vector $(0.23, 1.43, 0.58, 0.011)$ is the diagonal of the covariance matrix for θ obtained from a pilot run. For the pMALA algorithm the proposal is scaled as $\sigma_\epsilon^2 = (0.23, 1.43, 0.58, 0.011)/4^{1/3}$.

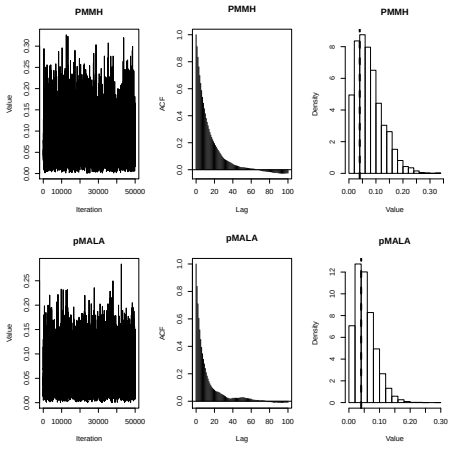

Figure 2: GARCH example. Trace plots, autocorrelation plots and posterior density (dashed line indicates true parameter) plots of the parameter γ from the MCMC sampler using PMMH and pMALA.

The trace plots given in Figure 2 show that the MCMC sampler mixes well for

both PMMH and pMALA. The posterior provides a good approximation of the parameter γ with the mode of the density matching the true parameter value. The autocorrelation plots show the reduced lagged correlation and improved mixing of the MCMC sampler using the pMALA algorithm compared to PMMH.

		PMMH	pMALA
Acc. rate		0.15	0.33
Ineff	$\text{logit}(\phi)$	35.01	**26.72**
	$\log(\mu)$	**38.42**	38.61
	$\text{logit}(\gamma)$	30.50	**21.24**
	$\log(\tau)$	32.80	**25.28**
SJD (x10^{-2})	$\text{logit}(\phi)$	3.38	**3.79**
	$\log(\mu)$	1.13	**1.44**
	$\text{logit}(\gamma)$	10.06	**12.47**
	$\log(\tau)$	0.17	**0.18**

Table 3: GARCH example. Comparison of the inefficiency and squared jump distance of PMMH and pMALA. Bold font indicates the best algorithm in terms of inefficiency and squared jump distance for each parameter.

The MCMC simulation results summarised in Table 3 show the significant improvement of pMALA over PMMH in terms of efficiency (except one parameter) and squared jump distance. Both metrics indicate that pMALA improves the mixing of the MCMC sampler by proposing new parameter values which are in the direction of the mode of the posterior. This increases the acceptance rate of the MCMC scheme as the sampler is less likely to become stuck in the tails of the density where new samples are unlikely to be accepted.

4.3 Stochastic volatility with leverage

The univariate stochastic volatility model is a state-space model with non-Gaussian observations, where the latent volatility follows an autoregressive process (see Shephard (2005) for a book length review). Variations of the stochastic volatility model have been extensively applied to model stock market returns. In this example we shall consider the stochastic volatility model with leverage given by Omori et al. (2007),

$$\begin{aligned} y_t &= \exp(x_t/2)\epsilon_t, \\ x_t &= \mu + \phi(x_{t-1} - \mu) + \eta_t, \\ x_0 &\sim \mathcal{N}(0, \sigma^2/(1-\phi^2)), \end{aligned}$$

where,

$$\begin{pmatrix} \epsilon_t \\ \eta_t \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho\sigma \\ \rho\sigma & \sigma^2 \end{pmatrix} \right).$$

The observations y_t are the returns, x_t is the latent log-volatility, μ is the drift, σ^2 is the volatility of the log-volatility and ϕ is the persistence parameter. This model also allows the errors in the observation and state transition equations to be correlated through the parameter ρ. In the context of stock market data

a negative value of ρ corresponds to an increase in volatility which follows from a drop in returns (Yu, 2005).

We apply the PMMH and pMALA algorithms to estimate the parameters $\theta = (\mu, \phi, \sigma, \rho)$, where the likelihood and score vector are obtained from a particle filter using 1000 particles. We use daily returns data from the S&P 500 index taken from January 1980 to December 1987 (2022 observations). This dataset has previously been studied by Yu (2005) using MCMC methods and by Jungbacker and Koopman (2007) using a Monte Carlo likelihood method.

The prior distributions for the parameters are: $\mu \sim \mathcal{N}(0, 1)$, $(\phi + 1)/2 \sim Beta(20, 1.5)$, $\sigma^2 \sim \mathcal{IG}(2.5, 0.025)$ and $\rho \sim U(-1, 1)$. The constrained parameters ϕ, ρ and $\sigma > 0$ are transformed to unconstrained parameters $\text{logit}(\phi)$, $\text{atanh}(\rho)$ and $\log(\sigma)$, where the Jacobian of the transformation is included in the MH acceptance ratio.

The RWM proposal of the PMMH algorithm is scaled as $\sigma_\epsilon^2 = (2.562)^2(0.017, 0.18, 0.037, 0.02)/4$, where the vector $(0.017, 0.18, 0.037, 0.02)$ is the diagonal of the covariance matrix for θ obtained from a pilot run. For the pMALA proposal the step-size is scaled as $\sigma_\epsilon^2 = (0.017, 0.18, 0.037, 0.02)/4^{1/3}$.

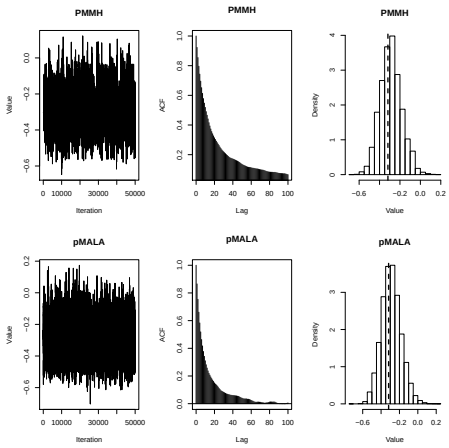

Figure 3: Stochastic Volatility example. Trace plots, autocorrelation plots and posterior density plots of the parameter ρ from the MCMC sampler using PMMH and pMALA. The dashed line indicates the posterior mean given by Yu (2005).

As with the previous examples Figure 3 displays the mixing of the MCMC samplers as well as the autocorrelation and posterior density plots. Both proposals show good mixing and sensible posteriors which contain the parameter estimates given by Yu (2005) and Jungbacker and Koopman (2007). The lagged correlation of the Markov chain given by the autocorrelation plots shows that, compared to PMMH, pMALA explores the posterior density more efficiently.

Table 4 provides a comparison of the two proposals in terms of their inefficiency and squared jump distance. For all parameters of the stochastic volatility model pMALA creates a more efficient MCMC sampler than PMMH. The increased acceptance rate indicates that pMALA allows the sampler to propose samples which are more likely to be accepted and therefore better explore the

		PMMH	pMALA
Acc. rate		0.10	0.21
Ineff	μ	57.02	**54.65**
	$\text{logit}(\phi)$	68.80	**51.92**
	$\log(\sigma)$	57.68	**40.10**
	$\text{atanh}(\rho)$	43.44	**37.28**
SJD ($\text{x}10^{-3}$)	μ	1.82	**1.90**
	$\text{logit}(\phi)$	16.27	**16.73**
	$\log(\sigma)$	3.25	**3.54**
	$\text{atanh}(\rho)$	1.64	**1.78**

Table 4: Stochastic volatility example. Comparison of the inefficiency and squared jump distance of PMMH and pMALA. Bold font indicates the best algorithm in terms of inefficiency and squared jump distance for each parameter.

posterior. The result is an increased squared jump distance between samples and reduced correlation between samples.

5 Discussion

The particle MALA proposal presented in this paper shows a significant improvement over the standard random walk Metropolis proposal when applied to the particle marginal Metropolis Hastings algorithm. One of the main advantages of this algorithm is its fast computational time, where the order of computation is equivalent to the computational effort required to estimate the likelihood. This means that more particles can be used to estimate the score vector and likelihood, resulting in estimates with lower variance and improved mixing of the MCMC sampler.

We have shown that it is possible to create MALA proposals using score estimates given by the $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ algorithms of Poyiadjis et al. (2011). Compared to the $\mathcal{O}(N^2)$ algorithm, our pMALA algorithm offers significant improvements in terms of computational cost. The benefit of this computational saving is highly significant as the pMALA algorithm can be executed with a larger number of particles, thus reducing the variance of the likelihood estimate. As shown, this reduction in variance considerably improves the efficiency of the MCMC sampler. For the Poyiadjis $\mathcal{O}(N)$ algorithm, our particle MALA algorithm is more efficient when implemented with equal computational cost. The improvement of pMALA also increases as the size of the dataset increases due to the quadratically increasing variance of the Poyiadjis $\mathcal{O}(N)$ algorithm. For the example given here, pMALA was up to 20% more efficient.

This proposal can also be used with more complex models where the derivative of the log posterior is not available for all parameters, but can be computed for a subset of the parameters. This will improve the overall efficiency of the sampler as pMALA will sample, from this subset, parameters more likely to be accepted. The remaining parameters can be sampled with the random walk Metropolis proposal.

A second order MALA proposal (Dahlin et al., 2013b) which takes account of the curvature of the posterior could be found using an estimate of the observed

information matrix. In principle, this would improve the mixing of the MCMC sampler by taking account of the local parameter covariance structure. However, estimates of the observed information matrix are not guaranteed to be positive definite which is an issue that would need to be addressed.

An important extension to this work would be to develop theoretical results establishing the optimal acceptance rate for pMALA. Recent results (Sherlock et al., 2013) have established optimal acceptance rates for the random walk Metropolis proposal which are helpful when tuning these proposals. Similar results for pMALA would make it easier to implement and reduce the time spent tuning the algorithm.

A Importance proposals

The importance proposals $\sum_{i=1}^{N} \xi_t^{(i)} q(x_t|x_{t-1}^{(i)}, y_t, \theta)$ for each example in Section 4 are given below.

Linear Gaussian model. For this model the fully adapted filter is available where x_t is sampled from the optimal proposal $q(x_t|x_{t-1}, y_t, \theta) = p(x_t|x_{t-1}, y_t, \theta)$ (see Doucet et al. (2000) for details) and the normalised posterior weights $w_t^{(i)} = 1/N$ are all equal. Explicitly the importance proposal is

$$\begin{aligned} \xi_t^{(i)} &\propto w_{t-1}^{(i)} \mathcal{N}(\alpha + \beta(\mu + \phi x_{t-1}^{(i)}), \beta^2\sigma^2 + \tau^2) \\ q^{\text{opt}}(x_t|x_{t-1}^{(i)}, y_t, \theta) &= \mathcal{N}(\Omega_t(\beta(y_t - \alpha)\tau^{-2} + (\mu + \phi x_{t-1}^{(i)})\sigma^{-2}), \Omega_t) \end{aligned}$$

where $\Omega_t = (\sigma^{-2} + \beta^2\tau^{-2})^{-1}$.

GARCH model. We again use the fully adapted filter for the GARCH model with noise where the importance proposal is

$$\begin{aligned} \xi_t^{(i)} &\propto w_{t-1}^{(i)} \mathcal{N}(0, \sigma_t^{2^{(i)}} + \tau^2) \\ q^{\text{opt}}(x_t|x_{t-1}^{(i)}, y_t, \theta) &= \mathcal{N}(\Omega_t y_t \tau^{-2}, \Omega_t) \end{aligned}$$

and $\Omega_t = (\sigma_t^{-2^{(i)}} + \tau^{-2})^{-1}$.

Stochastic volatility model with leverage. We use the importance proposal described by Omori et al. (2007),

$$\begin{aligned} \xi_t^{(i)} &\propto w_{t-1}^{(i)} \mathcal{N}(0, \exp(\mu_t^{(i)})) \\ q(x_t|x_{t-1}^{(i)}, y_t, \theta) &= \mathcal{N}(\mu_t^{(i)}, (1-\rho^2)\sigma^2) \end{aligned}$$

where $\mu_t^{(i)} = \mu + \phi(x_{t-1}^{(i)} - \mu) + \rho\sigma \exp(-x_{t-1}^{(i)}/2) y_t$.

References

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342.

Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725.

Beaumont, M. (2003). Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–60.

Bollerslev, T., Engle, R., and Nelson, D. (1994). ARCH models. In Engle, R. and McFadden, D., editors, *Handbook of Econometrics*, volume IV, chapter 49, pages 2959–3028. Elsevier.

Cappé, O., Godsill, S., and Moulines, E. (2007). An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924.

Cappé, O., Moulines, E., and Ryden, T. (2005). *Inference in Hidden Markov Models*. Springer Series in Statistics. Springer.

Crisan, D. and Doucet, A. (2002). A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746.

Dahlin, J., Lindsten, F., and Schon, T. (2013a). Particle Metropolis Hastings Using Langevin Dynamics. In *Proceedings of the 38th International Conference on Acoutics and Speech and Signal Processing*.

Dahlin, J., Lindsten, F., and Schön, T. (2013b). Second-order Particle MCMC for Bayesian Parameter Inference. *arXiv:1311.0686v1*.

Del Moral, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and Its Applications. Springer.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, 10(3):197–208.

Durbin, J. and Koopman, S. (2001). *Time Series Analysis by State Space Methods*. Oxford Statistical Science Series. Oxford University Press.

Fearnhead, P. (2011). MCMC for state-space models. In Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*, pages 513–529. Chapman and Hall.

Fearnhead, P., Wyncoll, D. P., and Tawn, J. (2010). A sequential smoothing algorithm with linear computational cost. *Biometrika*, 97(2):447–464.

Girolami, M. and Calderhead, B. (2011). Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear and linear Bayesian state estimation. *IEE Proceedings*, 140(2):107–113.

Jungbacker, B. and Koopman, S. J. (2007). Monte Carlo Estimation for Non-linear Non-Gaussian State Space Models. *Biometrika*, 94(4):827–839.

Nemeth, C., Fearnhead, P., and Mihaylova, L. (2013). Particle approximations of the score and observed information matrix for parameter estimation in state space models with linear computational cost. *arXiv:1306.0735v1*.

Omori, Y., Chib, S., Shephard, N., and Nakajima, J. (2007). Stochastic volatility with leverage: Fast and efficient likelihood inference. *Journal of Econometrics*, 140(2):425–449.

Pitt, M., Silva, R., Giordani, P., and Kohn, R. (2012). On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151.

Pitt, M. K. (2002). Smooth Particle Filters for Likelihood Evaluation and Maximisation. Technical Report 651, Warwick University.

Pitt, M. K. and Shephard, N. (1999). Filtering via Simulation: Auxiliary Particle Filters. *Journal of the American Statistical Association*, 94(446):590–599.

Poyiadjis, G., Doucet, A., and Singh, S. S. (2011). Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80.

Roberts, G. and Tweedie, R. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363.

Roberts, G. O., Gelman, A., and Gilks, W. (1997). Weak Convergence and Optimal Scaling of the Random Walk Metropolis Algorithms. *The Annals of Applied Probability*, 7(1):110–120.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268.

Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367.

Shephard, N. (2005). *Stochastic volatility: selected readings*. Advanced texts in econometrics. Oxford University Press.

Sherlock, C., Thiery, A. H., Roberts, G. O., and Rosenthal, J. S. (2013). On the efficiency of pseudo-marginal random walk Metropolis algorithms. *arXiv:1309.7209v1*.

West, M. (1993). Approximating posterior distributions by mixture. *Journal of the Royal Statistical Society. Series B*, 55(2):409–422.

Yu, J. (2005). On leverage in a stochastic volatility model. *Journal of Econometrics*, 127(2):165–178.